

WHAT IS CLAIMED IS:

- 1           1.       A method for communicating with a device, comprising:  
2           executing a kernel module in a memory;  
3           executing at least one kernel thread in the memory to handle calls to device driver  
4 functions for the kernel module; and  
5           executing, with the at least one kernel thread, calls to device driver functions for  
6 the kernel module running in a kernel context.
  
- 1           2.       The method of claim 1, wherein the kernel module spawns at least one  
2 kernel thread to execute the calls to the device driver functions for the kernel module.
  
- 1           3.       The method of claim 1, further comprising:  
2           accessing, with one kernel thread, device information from the device; and  
3           buffering the accessed device information.
  
- 1           4.       The method of claim 3, wherein a kernel module function requests device  
2 information, further comprising:  
3           in response to the request for the device information, accessing the buffered  
4 device information.
  
- 1           5.       The method of claim 1, wherein the kernel thread accesses buffered device  
2 information periodically and independently of kernel module requests for the device  
3 information.
  
- 1           6.       The method of claim 1, further comprising:  
2           buffering a parameter list;  
3           setting device parameters in the buffered parameter list to values provided by  
4 kernel module functions.

- 1           7.     The method of claim 6, further comprising:  
2           setting a flag indicating that the kernel thread needs to set parameters at the  
3     device to device parameter values set in the parameter list.
  
- 1           8.     The method of claim 6, further comprising:  
2           spawning a kernel thread to set device parameters to parameter values buffered in  
3     the parameter list.
  
- 1           9.     The method of claim 7, wherein the kernel thread spawned to set device  
2     parameter values processes the parameter list to locate buffered parameter values and set  
3     the device parameters to the buffered parameter values.
  
- 1           10.    The method of claim 7, wherein the kernel thread processes the parameter  
2     list by further performing:  
3           applying a lock on information in the parameter list including the located buffered  
4     parameter values;  
5           after applying the lock, copying the parameter values from the parameter list to a  
6     temporary buffer, wherein the device parameters are set to the parameter values from the  
7     parameter list in the temporary buffer; and  
8           releasing the lock after copying the parameter values from the parameter list to  
9     the temporary buffer.
  
- 1           11.    The method of claim 10, further comprising:  
2           disabling higher priority contexts before locking the parameter list; and  
3           enabling the higher priority contexts after releasing the lock on the parameter list.
  
- 1           12.    The method of claim 11, wherein the higher priority context comprises a  
2     bottom half or Interrupt Request (IRQ) context.

1           13.    The method of claim 10, further comprising:  
2           after releasing the lock, executing device driver functions to configure the device  
3           with the parameter values in the temporary buffer.

1           14.    The method of claim 1, further comprising:  
2           initiating, with the kernel module, an access request with respect to device  
3           information;  
4           disabling any higher priority contexts capable of accessing the device  
5           information;  
6           obtaining a lock for the device information subject to the access request;  
7           providing the kernel module access to the device information;  
8           releasing the lock; and  
9           enabling all higher priority contexts that were disabled.

1           15.    A system, comprising:  
2           a network device;  
3           a memory;  
4           a processor executing code to perform:  
5                (i) execute a network device driver in memory to control the network  
6           device;  
7                (ii) execute a kernel module in the memory;  
8                (iii) execute at least one kernel thread in the memory to handle calls to  
9           device driver functions for the kernel module; and  
10               (iv) execute, with the at least one kernel thread, calls to device driver  
11           functions for the kernel module running in a kernel context.

1           16.    The system of claim 15, wherein the kernel module spawns at least one  
2   kernel thread to execute the called device driver functions.

1           17.    The system of claim 15, wherein the processor executes code to further  
2   perform:  
3           access, with one kernel thread, device information from the device; and  
4           buffer the accessed device information.

1           18.    The system of claim 17, wherein a kernel module function requests device  
2   information, wherein the processor executes the code to further perform:  
3           in response to the request for the device information, accessing the buffered  
4   device information.

1           19.    The system of claim 15, wherein the kernel thread accesses device  
2   information periodically and independently of kernel module requests for device  
3   information.

1           20.    The system of claim 15, wherein the processor executes the code to further  
2   perform:  
3           buffering a parameter list; and  
4           setting device parameters in the buffered parameter list to values provided by  
5   kernel module functions.

1           21.    The system of claim 20, wherein the processor executes the code to further  
2   perform:  
3           setting a flag indicating that the kernel thread needs to set parameters at the  
4   device to device parameter values set in the parameter list.

1           22.     The system of claim 21, wherein the kernel thread spawned to set device  
2     parameter values processes the parameter list to locate buffered parameter values and set  
3     the device parameters to the buffered parameter values.

1           23.     The system of claim 21, wherein the executed kernel thread processes the  
2     parameter list by further performing:  
3             applying a lock on information in the parameter list including the located buffered  
4     parameter values;  
5             after applying the lock, copying the parameter values from the parameter list to a  
6     temporary buffer, wherein the device parameters are set to the parameter values from the  
7     parameter list in the temporary buffer; and  
8             releasing the lock after copying the parameter values from the parameter list to  
9     the temporary buffer.

1           24.     The system of claim 23, wherein the processor executes the code to further  
2     perform:  
3             disabling higher priority context before locking the parameter list; and  
4             enabling the higher priority contexts after releasing the lock on the parameter list.

1           25.     The system of claim 23, wherein the processor executes the code to further  
2     perform:  
3             after releasing the lock, executing device driver functions to configure the device  
4     with the parameter values in the temporary buffer.

1           26.     The system of claim 15, wherein the processor executes the code to further  
2     perform:  
3             initiating, with the kernel module, an access request with respect to device  
4     information;

5           disabling any higher priority contexts capable of accessing the device  
6   information;  
7           obtaining a lock for the device information subject to the access request;  
8           providing the kernel module access to the device information;  
9           releasing the lock; and  
10          enabling all higher priority contexts that were disabled.

1           27.    An article of manufacture for communicating with a device, wherein the  
2   article of manufacture causes operations to be performed, the operations comprising:  
3           executing a kernel module;  
4           executing at least one kernel thread to handle calls to device driver functions for  
5   the kernel module; and  
6           executing, with the at least one kernel thread, calls to device driver functions for  
7   the kernel module running in a kernel context.

1           28.    The article of manufacture of claim 27, wherein the kernel module spawns  
2   at least one kernel thread to execute the called device driver functions.

1           29.    The article of manufacture of claim 27, wherein the operations further  
2   comprise:  
3           accessing, with one kernel thread, device information from the device; and  
4           buffering the accessed device information.

1           30.    The article of manufacture of claim 29, wherein a kernel module function  
2   requests device information, wherein the operations further comprise:  
3           in response to a request for the device information, accessing the buffered device  
4   information.

1           31.     The article of manufacture of claim 27, wherein the kernel thread accesses  
2 buffered device information periodically and independently of kernel module requests for  
3 device information.

1           32.     The article of manufacture of claim 27, wherein the operations further  
2 comprise:  
3           buffering a parameter list;  
4           setting device parameters in the buffered parameter list to values provided by  
5 kernel module functions.

1           33.     The article of manufacture of claim 32, wherein the operations further  
2 comprise:  
3           setting a flag indicating that the kernel thread needs to set parameters at the  
4 device to device parameter values set in the parameter list.

1           34.     The article of manufacture of claim 27, wherein the kernel thread spawned  
2 to set device parameter values processes the parameter list to locate buffered parameter  
3 values and set the device parameters to the buffered parameter values.

1           35.     The article of manufacture of claim 34, wherein the kernel thread  
2 processes the parameter list by further performing:  
3           applying a lock on information in the parameter list including the located buffered  
4 parameter values;  
5           after applying the lock, copying the parameter values from the parameter list to a  
6 temporary buffer, wherein the device parameters are set to the parameter values from the  
7 parameter list in the temporary buffer; and  
8           releasing the lock after copying the parameter values from the parameter list to  
9 the temporary buffer.

1           36.    The article of manufacture of claim 35, wherein the operations further  
2   comprise:  
3            disabling higher priority contexts before locking the parameter list; and  
4            enabling the higher priority contexts after releasing the lock on the parameter list.

1           37.    The article of manufacture of claim 36, wherein the higher priority context  
2   comprises a bottom half or Interrupt Request (IRQ) context.

1           38.    The article of manufacture of claim 35, wherein the operations further  
2   comprise:  
3            after releasing the lock, executing device driver functions to configure the device  
4   with the parameter values in the temporary buffer.

1           39.    The article of manufacture of claim 27, wherein the code executes  
2   operations to further perform:  
3            initiating, with the kernel module, an access request with respect to device  
4   information;  
5            disabling any higher priority contexts capable of accessing the device  
6   information;  
7            obtaining a lock for the device information subject to the access request;  
8            providing the kernel module access to the device information;  
9            releasing the lock; and  
10          enabling all higher priority contexts that were disabled.